UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/838,078	04/19/2001	Shubhendu S. Mukherjee	1662-37200 JMH (P00-3159	9585

22879 7590 08/24/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
O BRIEN, BARRY J	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 08/24/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/838,078	MUKHERJEE ET AL.	
	Examiner	Art Unit	
	Barry J. O'Brien	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 June 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-21 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed on record in the file: Amendment A as received on 6/22/2004 and Change of Address as received on 6/22/2004.

Specification

3. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

5. Claims 1-13 and 15-20 are rejected under 35 U.S.C. 102(a) as being anticipated by Rotenberg, *AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors*.

6. Regarding claim 1, Rotenberg has taught a computer system, comprising:

a. A pipelined, simultaneous and redundantly threaded ("SRT") processor (see Col.4 lines 11-38) comprising a fetch unit that further comprises a branch predictor (see Col.6 lines 4-17 and Col.10 lines 20-27) and a counter, said counter causes instruction in a trailing thread to be fetched after corresponding instructions in a leading thread are fetched. Here, while not taught explicitly, there is inherently some sort of counter to keep track of "the number of free entries" in the delay buffer (see Col.11 lines 39-45), because certain things must happen when the number of free entries reaches a certain point, specifically the stalling of the appropriate thread (see Col.11 lines 21-45). Thus, because the "number of free entries" determines if the trailing thread is stalled or not (i.e. fetching instructions for execution or not), the "counter" keeping track of the "number of free entries" causes instructions in the trailing thread that correspond to the leading thread to be fetched.

d. A main system memory coupled to said processor (see Col.4 lines 50-52 and Col.9 line 49 – Col.10 line14),

e. Wherein said SRT processor processes a set of instructions in the leading thread and also in the trailing thread (see Col.4 line 44 – Col.5 line 9) and the SRT processor speculates on the outcome of branch instruction in the leading thread using the branch predictor (see Col.6 lines 4-10), but wherein the SRT processor does not speculate on the outcome of branch instructions in the trailing thread and instead uses the actual outcome of branch instructions in the leading thread to

predict the outcome of corresponding branch instructions in the trailing thread (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20).

7. Regarding claim 2, Rotenberg has taught the computer system of claim 1, further comprising:

a. A branch outcome queue located in the fetch unit (see Col.4 lines 44-52),

b. Wherein the actual outcomes of branch instructions in the leading thread are placed in the branch outcome queue (see Col.4 lines 44-52, Col.6 lines 4-17, 24-33).

8. Regarding claim 3, Rotenberg has taught the computer system of claim 2, wherein the fetch unit accesses the branch outcome queue and not the branch predictor to predict the outcome of branch instructions in the trailing thread (see Col.6 lines 24-44).

9. Regarding claim 4, Rotenberg has taught the computer system of claim 2, wherein the branch instruction queue is a FIFO buffer (see Col.4 lines 44-52).

10. Regarding claim 5, Rotenberg has taught the computer system of claim 2, wherein the individual branch outcome entries in the branch outcome queue comprise a program type identifier (see Col.10 lines 8-14) and a target address for the location of the next instruction in the thread to be executed (see Col.4 lines 44-52). Here, the program type identifier is the thread identifier which allows different program threads to be distinguished from one another in the delay buffer, and the program counter updates that branch instructions created in the leading thread are target addresses to be executed by the trailing thread.

11. Regarding claim 6, Rotenberg has taught the computer system of claim 2 further comprising:

a. A register update unit (see "Issue Buffers" of Fig.4),

b. Wherein the register update unit is configured to hold instructions in a queue until the instructions are executed and retired by the SRT processor (see Fig.4),

c. Wherein the outcomes of branch instructions in the leading thread are not placed in the branch outcome queue until the branch instructions retire from the register update unit (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20). Here, instructions in the first thread are committed (retired) at the same time they are sent to the delay buffer for re-execution (see Fig.2, Col.4 lines 44-52 and Col.11 lines 6-10).

12. Regarding claim 7, Rotenberg has taught the computer system o claim 2, further comprising:

a. A slack counter located in the fetch unit (see Col.4 lines 44-52),

b. Wherein the slack counter is configured to maintain an approximately constant number of instructions of separation between corresponding instructions in the leading and trailing threads (see Col.11 lines 21-45). Here, the delay buffer controls the number of instructions in itself by stalling the appropriate thread to keep its value as close to a constant full as possible.

13. Regarding claim 8, Rotenberg has taught the computer system of claim 3, further comprising:

a. Wherein if the branch outcome queue becomes full, execution of instructions in the leading thread is temporarily halted to prevent more branch outcomes from entering the branch outcome queue (see Col.11 lines 21-45),

b. Wherein if the branch outcome queue becomes empty, execution of instructions in the trailing thread is temporarily halted to allow more branch outcomes to enter the branch outcome queue (see Col.11 lines 21-45).

14. Regarding claim 9, Rotenberg has taught a pipelined, simultaneous and redundantly threaded ("SRT") processor (see Col.4 lines 11-38), comprising:

a. A fetch unit that fetches instructions from a plurality of threads of instructions (see "Fetch" stage of Fig.5, Col.6 lines 4-17, Col.10 lines 20-27 and Col.11 lines 1-5),

b. A program counter configured to assign program counter values to instructions in each thread that are fetched by the fetch unit (see Col.10 lines 15-27),

c. An instruction cache coupled to said fetch unit for storing instructions to be decoded and executed (see "Trace Cache" of Figs.4,5 and Col.7 lines 21-36),

d. Decode logic coupled to said instruction cache to decode the type of instructions stored in said instruction cache (see "Decode" in "Dispatch" stage of Fig.5 and Col.11 lines 1-5),

e. A counter (see Col.11 lines 39-45),

f. Wherein said processor is configured to detect transient faults during program execution (see Col.2 lines 36-41) by executing instructions in at least two redundant copies of a program thread (see Col.4 line 44 – Col.5 line 9) and wherein misspeculation caused by incorrectly predicting the outcomes of branch instructions in a second program thread is avoided by using the actual outcomes of branch instructions in a first program thread to correctly determine the outcome

of branch instructions in the second program thread (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20).

g. Wherein said counter causes instructions in the second program thread to be fetched after corresponding instructions in the first program thread are fetched. Here, while not taught explicitly, there is inherently some sort of counter to keep track of "the number of free entries" in the delay buffer (see Col.11 lines 39-45), because certain things must happen when the number of free entries reaches a certain point, specifically the stalling of the appropriate thread (see Col.11 lines 21-45). Thus, because the "number of free entries" determines if the trailing thread is stalled or not (i.e. fetching instructions for execution or not), the "counter" keeping track of the "number of free entries" causes instructions in the trailing thread that correspond to the leading thread to be fetched.

15. Regarding claim 10, Rotenberg has taught the SRT processor of claim 9, wherein instructions in the first program thread execute in advance of the corresponding instructions in the second program thread thereby creating a slack of instructions between the first and second program threads and wherein said slack is sufficient to allow the SRT processor to resolve any misspeculation in the first program thread prior to providing correct branch outcome results to the second program thread (see Col.6 line 24 – Col.7 line 6).

16. Regarding claim 11, Rotenberg has taught the SRT processor of claim 10, wherein said fetch unit comprises:

a. A slack counter (see Col.4 lines 44-52) configured to maintain a target number of instructions of separation between corresponding instructions in the first and

second threads (see Col.11 lines 21-45). Here, the delay buffer controls the number of instructions in itself by stalling the appropriate thread to keep its value as close to a constant full as possible.

17. Regarding claim 12, Rotenberg has taught the SRT processor of claim 9, wherein said fetch unit comprises:

a. A branch predictor for predicting the outcomes of branch instructions in the first program thread (see Col.6 lines 4-10),

b. A branch outcome queue for storing the actual outcomes of branch instructions in the first program thread (see Col.4 lines 44-52),

c. Wherein the actual outcomes of branch instructions in the first program thread are stored in the branch outcome queue after the branch instructions in the first program thread are retired by the SRT processor (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20). Here, instructions in the first thread are committed (retired) at the same time they are sent to the delay buffer for re-execution (see Fig.2 and Col.4 lines 44-52).

d. Wherein the fetch unit uses the branch outcome queue and not the branch predictor to predict the outcomes of branch instructions in the second program thread (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20).

18. Regarding claim 13, Rotenberg has taught the SRT processor of claim 12, wherein the SRT processor is an out-of-order processor capable of executing instructions in the most efficient order, but wherein branch instructions are executed in the same order in both the first and second program threads. Here, the processor can issue and execute multiple instructions from the same

thread in parallel (out-of-order) (see Figs. 4 and 5), but executes a branch instruction in order in the leading thread so as to deal with all control dependencies so that the same branch instruction can be executed without having to wait for those dependencies in the trailing thread (see Col.5 line 33 -Col.6 line 17 and Col.6 lines 24-44).

19. Regarding claim 15, Rotenberg has taught the SRT processor of claim 12, wherein the individual outcomes stored in the branch outcome queue comprise:

a. A program type classifying the branch instruction (see Col.10 lines 8-14),

b. A target address corresponding to the instruction to be executed immediately following the branch instruction (see Col.4 lines 44-52),

c. Wherein during execution of the second program thread, the SRT processor may identify the appropriate branch instruction using the program counter value and may also fetch instructions ahead of the branch instruction using the target address (see Col.6 lines 4-10). Here, the program type identifier is the thread identifier which allows different program threads to be distinguished from one another in the delay buffer, and the program counter updates that branch instructions created in the leading thread are target addresses to be executed by the trailing thread (see Col.4 lines 44-52 and Col.6 lines 24-44).

20. Regarding claim 16, Rotenberg has taught the SRT processor of claim 12, further comprising:

a. Wherein if the branch outcome queue becomes full, the first thread is stalled to prevent more branch outcomes from entering the ranch outcome queue (see Col.11 lines 21-45),

b. Wherein if the branch outcome queue becomes empty, the second thread is stalled to allow more branch outcomes to enter the branch outcome queue (see Col.11 lines 21-45).

21. Regarding claim 17, Rotenberg has taught a method of predicting branch instructions in a simultaneous and redundantly threaded ("SRT") processor (see Col.4 lines 11-38) which can fetch and execute a set of instructions in two separate threads so that each thread includes substantially the same instructions as the other thread, one of said threads being a leading thread and the other of said threads being a trailing thread (see Col.4 line 44 – Col.5 line 9), the method comprising:

a. Training a branch predictor to store predicted outcomes from branch instructions in the leading thread (see Col.6 lines 4-10, Col.6 line 24 – Col.7 line 6 and Col.10 line 20),

b. Probing the branch predictor to predict outcomes of future executions of branch instructions in the leading thread (see Col.6 lines 4-10, Col.6 line 24 – Col.7 line 6 and Col.10 line 20),

c. Storing actual outcomes of branch instructions in the leading thread in a branch outcome queue (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20),

d. Probing the branch outcome queue to predict outcomes of corresponding branch instruction in the trailing thread (see Col.6 line 24 – Col.7 line 6 and Col.10 line 20).

e. Decrementing a counter when the leading thread commits an instruction and incrementing the counter when the trailing thread commits an instruction. Here,

while not taught explicitly, there is inherently some sort of counter to keep track of "the number of free entries" in the delay buffer (see Col.11 lines 39-45), because certain things must happen when the number of free entries reaches a certain point, specifically the stalling of the appropriate thread (see Col.11 lines 21-45). Here, instructions in the leading thread are considered committed when they are sent to the delay buffer (see Fig.2 and Col.4 lines 44-52), and instructions in the trailing thread are considered committed when its entry is removed from the delay buffer (see Fig.2 and Col.5 lines 1-9). Thus, committing instructions from the leading thread adds an entry to the delay buffer, thereby incrementing the "number of free entries", whereas committing instructions from the redundant thread removes an entry from the delay buffer, thereby decrementing the "number of free entries".

22. Regarding claim 18, Rotenberg has taught the method of claim 17 further comprising:

a. Executing the branch instructions in the leading and trailing threads in program order. Here, the processor can issue and execute multiple instructions from the same thread in parallel (out-of-order) (see Figs. 4 and 5), but executes a branch instruction in order in the leading thread so as to deal with all control dependencies so that the same branch instruction can be executed without having to wait for those dependencies in the trailing thread (see Col.5 line 33 -Col.6 line 17 and Col.6 lines 24-44).

23. Regarding claim 19, Rotenberg has taught the method of claim 18, further comprising:

a. Storing the actual outcomes of branch instructions in the leading thread in the branch outcome queue after the branch instructions retire (see Col.4 lines 44-52 and Col.6 lines 24-44),

b. Wherein the outcomes are identified by a branch identifier (see Col.10 lines 8-14) and a target address signifying the subsequent instruction to be executed as a result of the outcome of the execution of the branch instruction (see Col.4 lines 44-52). Here, the program type identifier is the thread identifier which allows different program threads to be distinguished from one another in the delay buffer, and the program counter updates that branch instructions created in the leading thread are target addresses to be executed by the trailing thread.

24. Regarding claim 20, Rotenberg has taught the method of claim 18, further comprising:

a. Using a FIFO buffer as the branch outcome queue (see Col.4 lines 44-52),

b. Wherein if the buffer becomes full, the leading thread is stalled to prevent more branch outcomes from entering the buffer (see Col.11 lines 21-45),

c. Wherein if the buffer becomes empty, the trailing thread is stalled to allow more branch outcomes to enter the buffer (see Col.11 lines 21-45).

Claim Rejections - 35 USC § 103

25. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

26. Claims 14 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, *AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors* as applied to claim 13 above, and further in view of Patterson et al., *Computer Organization & Design: The Hardware/Software Interface.*

27. Regarding claim 14, Rotenberg has taught the SRT processor of claim 13, wherein the branch outcome queue is a FIFO buffer (see Col.4 lines 44-52), as well as the SRT processor being concerned with providing a high level of fault tolerance (see Col.4 lines 11-42), but has not explicitly taught wherein data is transmitted to and from the buffer using an error correction technique.

28. However, Patterson has taught data being transferred to and from memory has the possibility of being corrupted and is thus transferred using some form of error correction (see Patterson p.B-34 and p.B-35). Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of Rotenberg to be more fault tolerant by using a form of error correction when transferring to the delay buffer.

29. Regarding claim 21, Rotenberg has taught the method of claim 18, wherein the processor is concerned with providing a high level of fault tolerance (see Col.4 lines 11-42), but has not explicitly taught wherein the method further comprises:

a. Transmitting data to and from the branch outcome queue using an error correction technique.

30. However, Patterson has taught data being transferred to and from memory has the possibility of being corrupted and is thus transferred using some form of error correction (see Patterson p.B-34 and p.B-35). Therefore, one of ordinary skill in the art would have found it

obvious to modify the processor of Rotenberg to be more fault tolerant by using a form of error correction when transferring to the delay buffer.

Response to Arguments

31. Applicant's arguments filed 6/22/2004 have been fully considered but they are not persuasive.

32. On page 11 of the present amendment, Applicant argues with respect to claims 1 and 9, in essence:

> *"Applicants amend claim 1 to specify that the SRT processor comprises a "counter" and that "said counter causes instructions in a trailing thread to be fetched after corresponding instruction in a leading thread." Rotenberg does not teach or even suggest the use of counter in this regard. At most, Rotenberg teaches a FIFO buffer (i.e. the delay buffer in Figure 2) to cause the R-Stream to lag behind the A-Stream."*

33. However, Rotenberg has taught the use of a counter in such a manner (see above paragraphs 6 and 14). While not taught explicitly, there is some sort of counter to keep track of "the number of free entries" in the delay buffer (see Col.11 lines 39-45), because certain things must happen when the number of free entries reaches a certain point, specifically the stalling of the appropriate thread (see Col.11 lines 21-45). Here entries in the delay buffer contain the program counter values that were executed in the leading thread and that are to be executed in the trailing thread (see Col.10 lines 20-27), and then the trailing thread uses those program counter values to execute an updated instruction stream (see Col.6 lines 24-33). Thus, because the "number of free entries" determines if the trailing thread is stalled or not (i.e. fetching

instructions for execution or not), the "counter" keeping track of the "number of free entries" causes instructions in the trailing thread that correspond to the leading thread to be fetched.

34. On page 12 of the present amendment, Applicant argues with respect to claim 17, in essence:

> "*Applicants amend method claim 17 to also require "decrementing a counter when the leading thread commits and instruction and incrementing the counter when the trailing thread commits an instruction." Rotenberg has no such teaching.*"

35. However, Rotenberg has taught a counter being used in such a manner (see above paragraph 21). While not taught explicitly, there is some sort of counter to keep track of "the number of free entries" in the delay buffer (see Col.11 lines 39-45), because certain things must happen when the number of free entries reaches a certain point, specifically the stalling of the appropriate thread (see Col.11 lines 21-45). Here, instructions in the leading thread are considered committed when they are sent to the delay buffer (see Fig.2 and Col.4 lines 44-52), and instructions in the trailing thread are considered committed when its entry is removed from the delay buffer (see Fig.2 and Col.5 lines 1-9). Thus, committing instructions from the leading thread adds an entry to the delay buffer, thereby incrementing the "number of free entries", whereas committing instructions from the redundant thread removes an entry from the delay buffer, thereby decrementing the "number of free entries".

Conclusion

36. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

37. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

38. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Barry J. O'Brien whose telephone number is (703) 305-5864. The examiner can normally be reached on Mon.-Fri. 6:30am-4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

39. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Barry J. O'Brien
Examiner
Art Unit 2183

BJO
8/20/2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100